Detailed Implementation Guidelines for the Interim Low Bit Rate Compression System

November 27, 1996

by

Austin Lan
(716) 253-5743
lana@kodak.com

Joe Reitz
(716) 253-6599
reitz@kodak.com

Eastman Kodak Company
Commercial and Government Systems
1447 St. Paul Street
Rochester, NY 14653-7118

1.0 Interim JPEG System Model

The Interim JPEG compression algorithm achieves very low bit rate compression using the scheme shown in Figure 1. Decimation of the original image is used to achieve bit rates beyond what JPEG can accomplish alone due to the fixed 8x8 block size encoding structure (0.5-0.8 bits/pixel). In this algorithm, the adverse effects of downsampling (e.g. aliasing and blurring) are traded-off with JPEG artifacts (e.g. blocking) by adjusting the relative compression contributions from each module. The quality of the reconstructed image after JPEG decompression and interpolation has been demonstrated to be competitive with several "state-of-the-art" low bit rate compression algorithms.

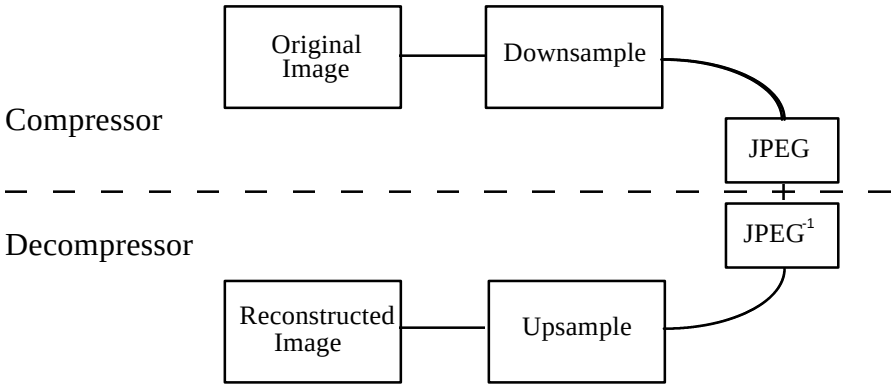

Figure 1: System model for Interim JPEG compression.

1.1 Downsampling Model

Conceptual downsamping is performed using the model shown in Figure 2. Note that all references to "x" refer to the *continuous spatial domain*. $H_d(j\Omega)$ represents the ideal downsampling filter that is applied to the digital data, T_o is the old sampling period (assume T_o=1), T is the new sampling period, and R is the downsample ratio. With T_o=1, the new sampling period, T, is equal to the downsample ratio, R. R is defined to be greater than one, so the new sampling rate is lower than the original.

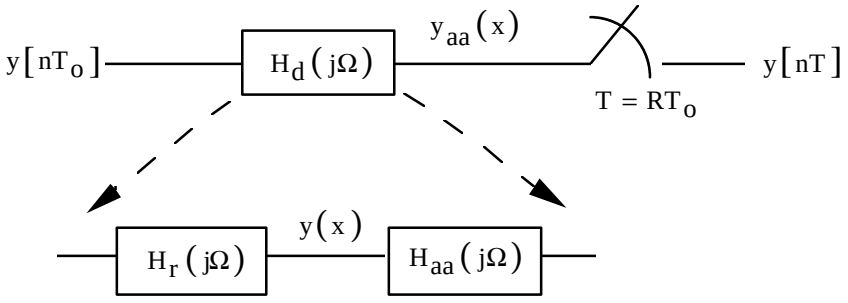

Figure 2: Downsampling model

$H_d(j\Omega)$ can conceptually be separated into two filters: an ideal reconstruction filter for discrete-to-continuous-spatial (D/C) conversion, and an ideal anti-aliasing filter. The reconstruction filter converts the digital signal to a continuous form, while the anti-aliasing filter prepares the continuous signal for resampling at a lower rate. The frequency responses of the two ideal filters are shown in Figure 3.

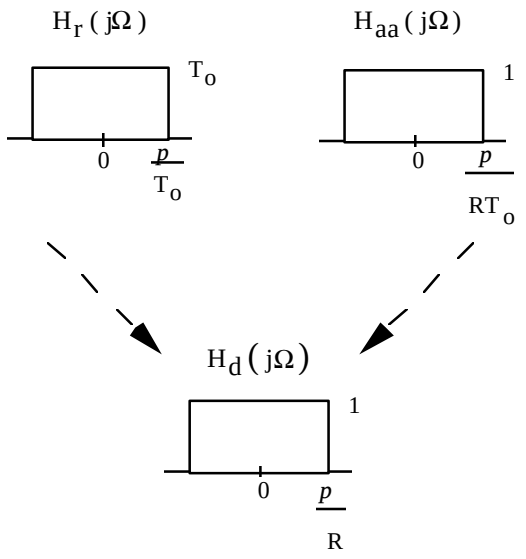

Figure 3: Frequency responses of the ideal downsampling filters.

The combined response, $H_d(j\Omega)$, is achieved by multiplying the reconstruction filter and anti-aliasing filter frequency responses (T_o=1). The resulting downsample filter performs both D/C conversion and anti-aliasing. The impulse response of the downsample filter is the sinc function defined in the following equation:

$$h_d(x) = \frac{1}{R}\operatorname{sinc}\left(\frac{1}{R}x\right) = \frac{\sin\left(\frac{p}{R}x\right)}{px} \qquad -\infty < x < \infty$$

The sampler shown in Figure 2 converts the continuous signal back to the digital domain, but at the new sampling rate. The sampling ratio, R, in this system can be any value greater than one, as opposed to discrete-time techniques for changing the sampling rate, which limit the sampling ratio to integer values.

1.1.1 Practial Considerations for Downsampling

The ideal filter can only be approximated due to its infinite length. The approximation is accomplished using a window of finite length to truncate the ideal impulse response. The window shape defines how abruptly the truncation is performed. This allows a trade-off between transition width and stopband attenuation of the windowed filter frequency response. The window to be used for downsampling is defined as follows:

$$w_d(x) = \begin{cases} \sqrt{\cos\left((p\,x)/\,flen\right)} & -flen/2 \le x \le flen/2 \\ 0 & else \end{cases}$$

where, *flen* is the downsample filter length defined as:

$$flen = base \cdot R$$

base refers to the desired base length for the downsample filter. This equation for filter length shows a dependency on the downsample ratio. This relationship gives the filter desirable properties which will be discussed in Section 2.1.

The windowed downsample filter is found by applying the window to the ideal impulse response:

$$h_{dwind}(x) = w_d(x) \times h_d(x) \qquad -\infty < x < \infty$$

The ideal impulse response and the truncating window are illustrated in Figure 4.

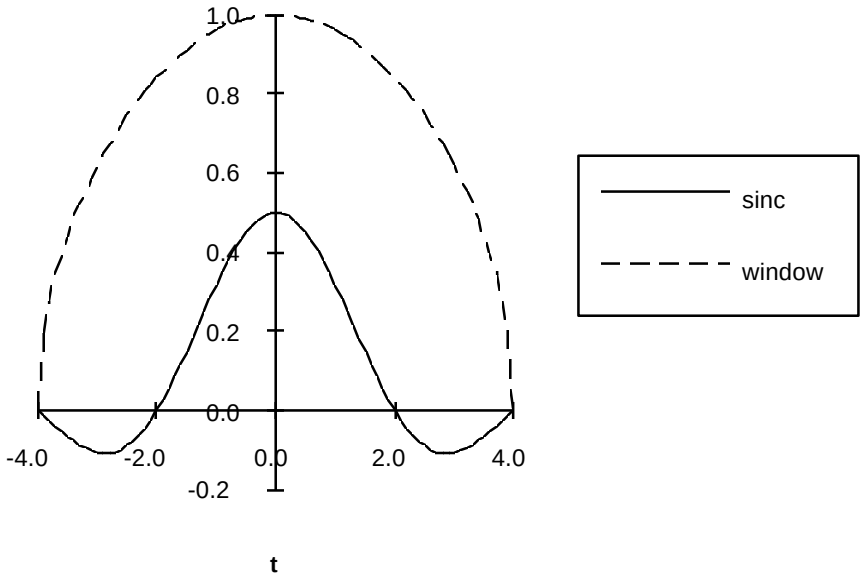

Figure 4: Ideal downsample filter and truncating window.

1.2 Upsampling Model

Figure 5 shows the conceptual upsampling process. $H_r(j\Omega)$ represents the ideal reconstruction filter that is used to perform D/C conversion. Once the signal is converted to the continuous domain, it is resampled with a new sampling period, T. The upsampling ratio, L, is defined to be greater than one, so the new sampling rate is higher than the original, T_o (assume T_o=1). The ideal frequency response is shown in Figure 6.

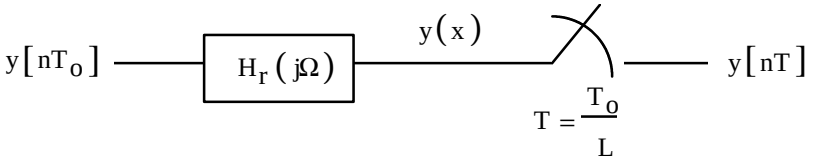

Figure 5: Upsampling model.

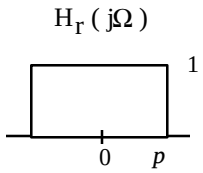

Figure 6: Frequency response of the ideal upsampling filter.

The impulse response of the upsample filter is the sinc function defined in the following equation:

$$h_r(x) = \mathrm{sinc}(x) = \frac{\sin(px)}{px} \qquad -\infty < x < \infty$$

1.2.1 Practical Considerations for Upsampling

Similar to the downsample case, the ideal upsample impulse response is truncated using a window in order to achieve a practical filter. The window to be used for upsampling is defined as follows:

$$w_u(x) = \begin{cases} \cos^2((px)/flen) & -flen/2 \le x \le flen/2 \\ 0 & else \end{cases}$$

where, *flen* is the desired filter length. The windowed upsampling filter is found by applying the window to the ideal impulse response:

$$h_{uwind}(x) = w_u(x) \times h_r(x) \qquad -\infty < x < \infty$$

The ideal impulse response and the truncating window are illustrated in Figure 7.

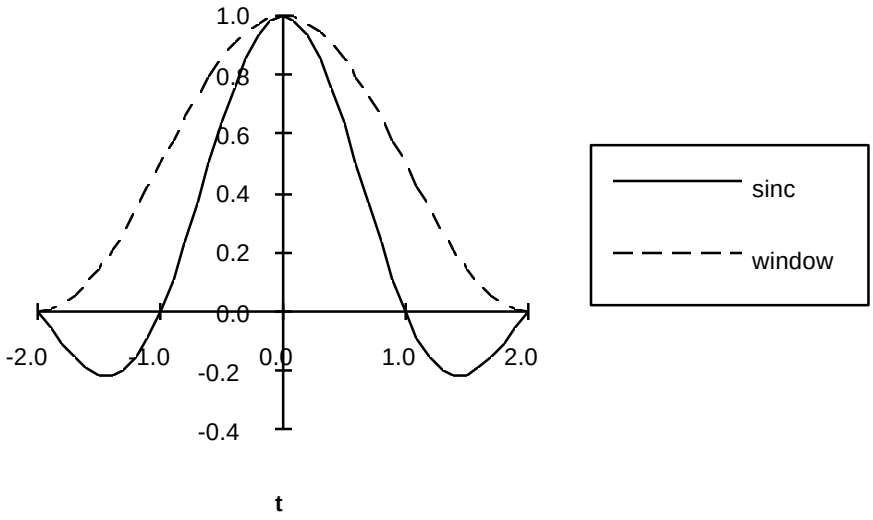

Figure 7: Ideal upsample filter and truncating window.

1.3 Frequency Response of Practical Filters

The following figures show the combined frequency response of downsample and upsample filter pairs for all *base* filter length combinations of 4 and 8. The downsample and upsample ratios for these examples are both 2.0. The actual filter length for the downsample filter is the base filter length multipled by the downsample ratio. The reasoning behind this is described in Section 2.0. The upsample filter length is the unmodified base length. Since the upsample filter in reality only influences the frequencies of the original signal in the interval [0, π/R], it is displayed in downsampled units so that its cutoff matches the downsample filter.

In general, the figures show that the chosen windows and filter lengths provide a sharper cutoff on the downsampling side, and a smoother response on the upsampling side. The windows described in Sections 1.1.1 and 1.2.1 form a matched pair due to their shape in the frequency domain. Gain from one window is matched with attenuation from the other to create a moderately flat joint response. The window choice for downsampling or upsampling is somewhat arbitrary, but intuition points to placing the filter with a sharp transition on the downsampling side to preserve as much of the original signal as possible. Image quality evaluations have verified this decision.

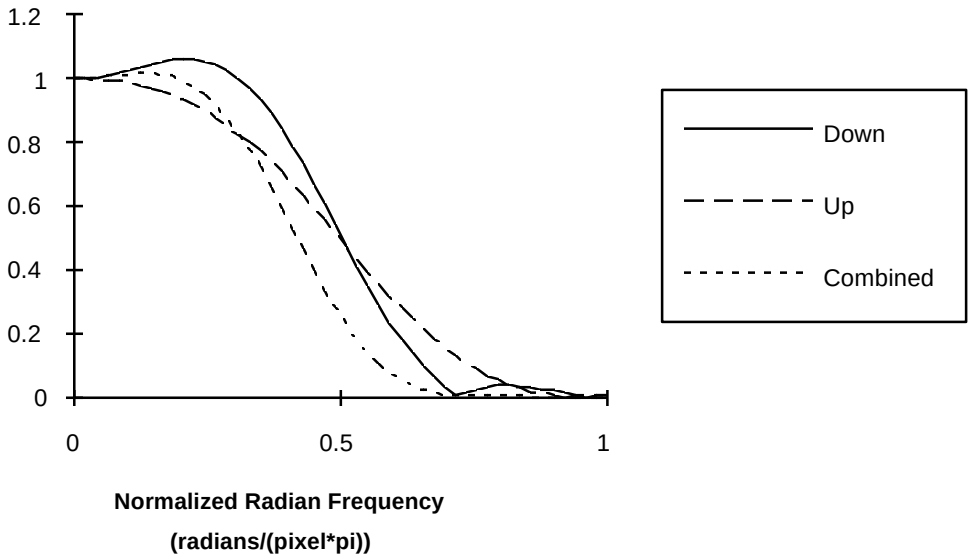

Figure 8: Downsample length 4; upsample length 4.

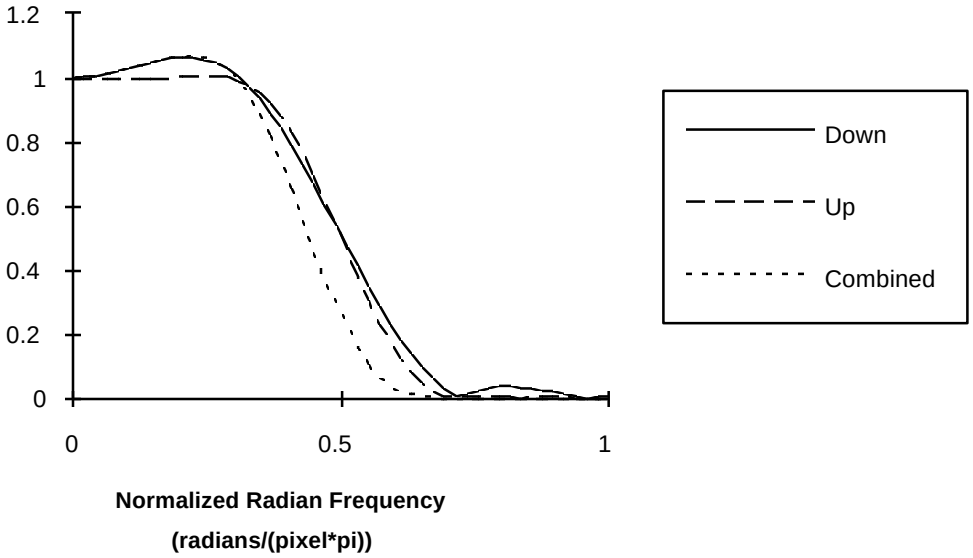

Figure 9: Downsample length 4; upsample length 8.

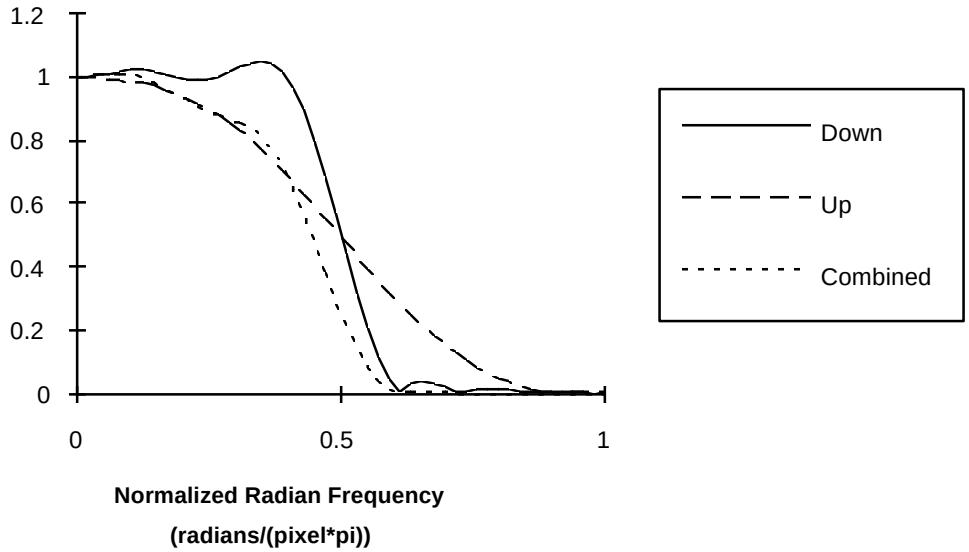

Figure 10: Downsample length 8; upsample length 4.

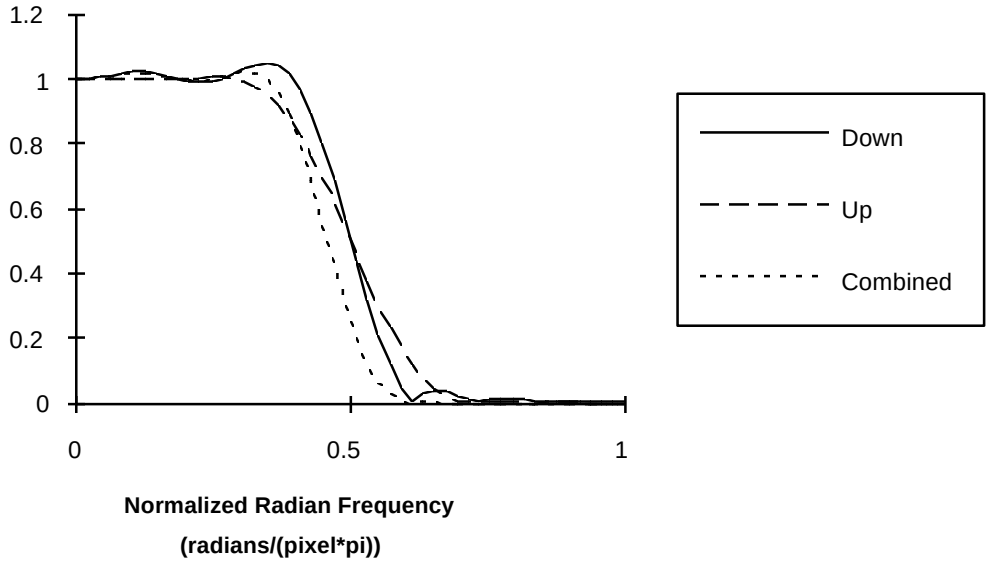

Figure 11: Downsample length 8; upsample length 8.

2.0 Filtering and Resampling Process

The downsample and upsample operations are simply convolutions using the kernels defined in Sections 1.1.1 and 1.2.1 followed by resampling. In two dimensions, downsampling is accomplished by performing the one-dimensional operation twice (e.g. downsample rows, then columns). The second downsampling should be performed on the **intermediate** result, which is the output from the first operation. Upsampling for the decoder in this system should reverse the processing order performed by the encoder (e.g. reconstruct columns first, then the rows), although it should not have significant impact if the order is not reversed. The choice of which dimension (rows or columns) to process first for the downsampling process should be driven by software optimization issues.

The convolution operation in Figures 2 and 5 is actually performed in the *continuous* spatial domain, as opposed to discrete-time convolution. The result after the convolution, but before the sampler, is always a continuous distribution of values. The sampler returns the signal to the discrete-time domain. The convolution and sampling operations can be combined to achieve a reduction in complexity by performing the necessary convolution calculations only at the locations that will be hit by the sampler. The other values from the convolution operation will never be used in later operations, so it is not necessary to consider them.

2.1 Details of the Downsampling Operation

The main input parameter into the downsampling module is the desired two-dimensional downsampling ratio for the entire image. This parameter is interpreted to mean the downsampled image is reduced in total size from the original by a factor that is approximately equal to the desired ratio. This ratio is only approximate since the ratio should be modified so that the downsampled image will have integer dimensions that are multiples of 8. If a dimension is not a multiple of 8, then the JPEG module must pad the image, which creates coding inefficiencies. The *desired* one-dimensional downsampling ratio that is applied to both the row and column dimensions is calculated by taking the square root of the input two-dimensional ratio. The *actual* one-dimensional downsampling ratio is calculated from the downsampled image dimensions, which are constrained to be multiples of 8 using the following equation:

$$downDim = \left\lfloor \frac{origDim}{8 * ratio} \right\rfloor * 8$$

where, *Dim* is either the rows or the columns, and *ratio* is the *desired* one-dimensional downsampling ratio. The *actual* downsampling ratio is then formed as the ratio of the original dimension to the downsampled dimension, so that it is greater than one. Note that the actual downsampling ratio can be different for each dimension. Both downsampling ratios must be calculated separately. The resulting ratio is equivalent to the new sampling period, T, discussed in Section 1.1.

The downsample filter length is a function of the downsampling ratio as mentioned in Section 1.3. It is calculated as the product of the downsampling ratio with the defined base length. The base length determines the non-zero support of the filter in terms of downsampled units. For example, a base length of 4 always spans 4 samples in the downsample domain regardless of the sampling ratio. The effect of this relationship in the frequency domain is to make the filter response much sharper as the downsampling ratio is increased. Sharper cutoff and lower stopband attenuation are issues that are much more critical when the desired cutoff of the lowpass filter decreases, since the signal energy tends to be concentrated in this region. The downsampled image is much more susceptible to aliasing and severe distortion when the signal energy is high in the filter cutoff region.

The mechanics of the filtering are shown pictorially in Figure 12 for a single dimension. Filtering in the other dimension is applied similarly. In the example presented, the sampling period is 2 (e.g. R=2 in Figures 2 and 3) for this dimension and the base filter length is 4. The pertinent values that need to be calculated for **every** output sample include: filter center, filter beginning, filter end, and filter coefficients. The filter center is indicated by the dashed line in the figure. Its value is calculated as:

$$center = iT_d + offset$$

where, i is an integer that represents the output sample index value and has values in the range [0..downDim], and T_d is the downsampling period (T_d>1). The offset term represents a shift that is applied to the intuitive filter center. The shift is used in order to avoid the situation where a natural zero-crossing of the sinc function falls on a non-zero input data sample. The filter coefficient is zero in this situation which wastes an operation and decreases the effective length of the filter. The offset that was used in our experimentation is defined as:

$$offset = 0.5 \cdot T_d - 0.5$$

Note that the **downsampled image** will have a shift corresponding to $offset/T_d$. This offset is removed by the upsampler defined in the next section. **Using a non-compliant upsampler (e.g. simple bilinear interpolation) will result in an image that is shifted by a value equal to *offset*.**

The filter beginning and end define where the filter is non-zero. These values can be used to determine the number of filter coefficients that need to be calculated, and the input data samples that fall within the filter's non-zero support. The beginning and end are calculated as:

$$begin = center - \frac{flen}{2} \qquad end = center + \frac{flen}{2}$$

where, *flen* is the actual downsample filter length.

The filter coefficients are calculated using the equations from Section 1.1. The number of coefficients needed for the current output sample is equal to the number of non-zero input data samples that fall within the non-zero support of the filter. It is necessary to determine the distance from a particular input data sample to the filter center in order to calculate the appropriate filter coefficient. This distance is labelled *delta* in Figure 12 as an example. The distance is calculated as:

$$delta = center - j$$

where, *j* is an integer value that represents a particular input data sample that falls within the filter's non-zero span. In Figure 12, the input data value that represents *j* is sample 3.

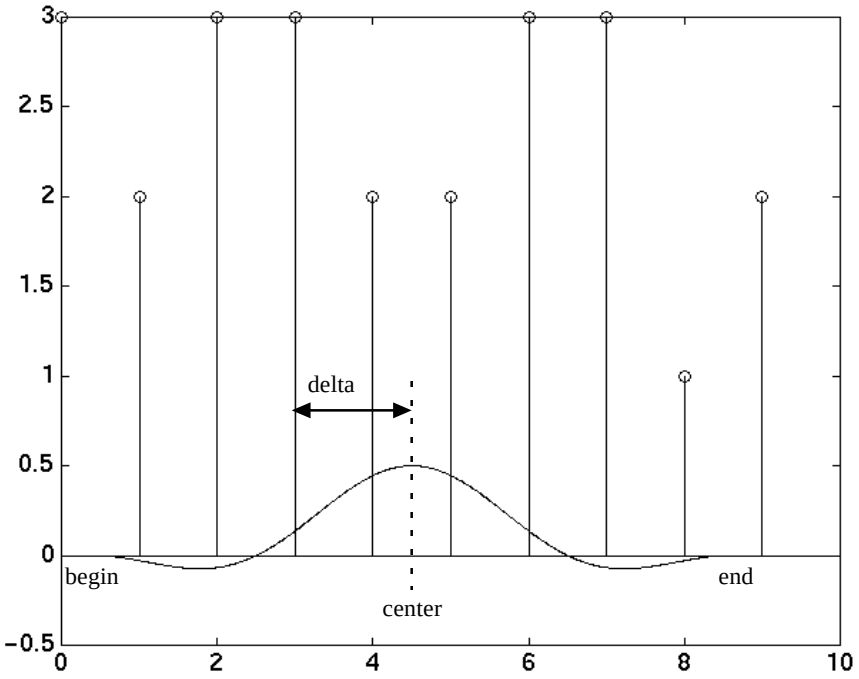

Figure 12: Pictorial view of the filtering operation.

The values for the example in Figure 12 using the equations described above are as follows:

- current output sample, $i = 2$
- downsample period, $T_d = 2$
- base filter length = 4
- actual filter length, *flen* = 8
- filter offset, $offset = 0.5 \cdot T_d - 0.5 = 0.5$
- filter center = 4.5
- filter beginning = 0.5
- filter end = 8.5
- delta for input sample 3, *delta* = 1.5
- filter coefficient corresponding to input sample 3:

$$coefficient = \sqrt{\cos\left((p \cdot delta)/\ flen\right)} \times \frac{\sin\left(\frac{p}{2} \cdot delta\right)}{p \cdot delta} = 0.137$$

Normalization of the filter coefficients is necessary in order to avoid uneven weightings for output samples. Normalization for this filtering operation is performed after all the filter coefficients are calculated for a single output sample. The values are then normalized to yield a unity gain DC frequency response using the following equation:

$$h_{normalized} = \frac{h_{original}}{\sum h_{original}}$$

where, $h_{original}$ represents the unmodified filter coefficients that are calculated. Note that this operation must be performed separately for every output sample.

In the course of downsampling an image in either dimension, the filter will span a region that includes values outside the input sampling grid. Mirroring of the input data should be used to fill in the needed values. The mirroring point coincides with the input data sample that is exactly on the edge (e.g. first sample in a row when padding on the left of the image). Therefore, the edge sample is never repeated. This is illustrated in Figure 13.

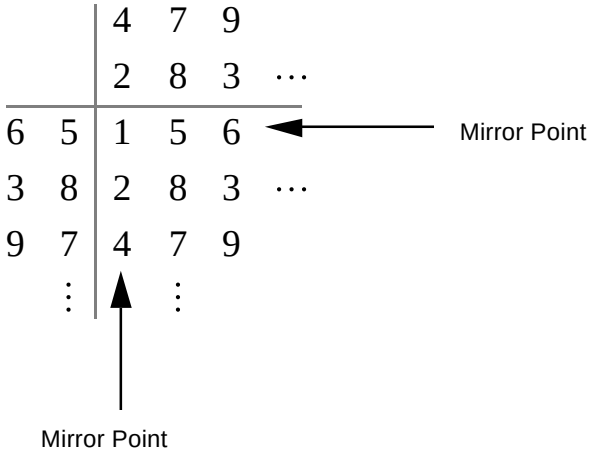

Figure 13: Illustration of mirroring for image edges. The bottom, right quadrant of values represents the top, left corner of the image.

2.2 Details of the Upsampling Process

The main inputs into the upsampling module is the target dimension of the upsampled image, and the dimensions of the original image. A distinction is made between the two sets of dimensions for the following discussion only. **In this compression system, the two sets of dimensions are constrained to be equivalent, so that the upsample period is equal to the inverse of the downsample period**

$$T_u = \frac{downDim}{origDim} = \frac{1}{T_d} < 1$$

where, downDim represents the dimensions of the downsampled image, and origDim represents the dimensions of the original image, or targeted upsampled image, since the values are equal. Similar to the downsampling operation, the upsampling period must be calculated separately for each dimension.

The upsample filter length is not a function of any sampling ratio, unlike the downsample filter length. The upsample filter length value describes how many non-zero samples will be used in interpolation of each output sample. For example, a filter length of 4 will effectively use 4 non-zero samples in the calculation of each output value.

The upsampling mechanics are similar to the downsampling process described in Section 2.1 when T_d is replaced by T_u. The exceptions are offset and filter center calculations. The offset in the downsampled image that must be removed is equivalent to offset/T_d as mentioned in Section 2.1. For example, the identified offset equation given in Section 2.1 is equal to the following in downsampled units:

$$offset_d = 0.5 - \frac{0.5}{T_d} = 0.5 - 0.5T_u$$

The latter equivalence follows from the constraint that $T_u = 1/T_d$ (e.g. The image is upsampled to the original image dimensions). Although this equality is forced upon these quantities, it is still possible to correctly handle the case where $T_u \_ 1/T_d$. For this situation, the downsampled image shift remains offset/T_d, and there exists no relationship with T_u.

The filter center calculation is modified to remove the shift in the downsampled image. The equation for the upsampler is as follows:

$$center = iT_u - offset_d$$

Normalization of the filter coefficients and edge padding are both performed the same as in the downsampling process.

3.0 Implementation and Testing Details

Several issues still exist in the design of this compression system. Most of the issues will be finalized when the trade-off between decimation and JPEG quantization is fully characterized. One such issue is the downsample and upsample filter lengths. *Base* filter length combinations that are recommended for testing are (4,4), (4,8), and (8,8). Figure 10, which corresponds to the (8,4) pair, shows that significant attenuation occurs over a large range of frequencies, similar to (4,4). The (8,4) pair, indeed, yielded images close in quality to that of (4,4) in experiments. Since the actual filter lengths that will be used in the system may be altered from these recommended values, it will be prudent to leave them open to change. It is not clear whether the extra complexity of a large filter (e.g. time to encode and decode will be longer) will be desired by end-users. There is an observable gain in quality, however, when the reconstructed image is viewed on a high resolution monitor. Leaving the values open to change will allow the developer to make this trade-off between complexity and quality.

The image should be processed twice in the course of downsampling (e.g. once in each dimension) as outlined in Section 2.0. The issue of which dimension to start with should be decided upon by weighing the software advantages and disadvantages. Should starting with rows first then columns have significant benefits to the software design, then this order can be chosen. The upsampler is defined to reverse the processing order in Section 2.0. Once again, this guideline can be redefined if the software can be enhanced by processing using the other ordering.

The offset that is used to shift the filter center must also remain open to change. The recommended value is given in Section 2.1, but repeated here for convenience:

$$offset = 0.5 \cdot T_d - 0.5$$

We will be looking at the implications of faster upsampling methods (e.g. bilinear interpolation) in the coming months, since there exists dedicated hardware to accomplish this task. The hardware will not be able to accomodate the offset that is applied during the filtering process. Therefore, it may be necessary to change the offset to zero, should fast upsampling provide acceptable image quality when compared with the upsampling method described in Section 2.2.

Another issue is that the software must be able to handle images that have bit depths ranging from 8 to 12 bits/pixel. It is possible for the system to map downsampled images that have a bit depth lower than 12 bits/pixel to the maximum 12-bit range that the JPEG standard allows. However, much of the increase in precision is lost once JPEG is applied, since the quantization for the 12-bit images will be much higher. Therefore, we recommended that the bit depth within the entire system remain at the native value of the input image (e.g. 8 bits/pixel should remain 8 bits/pixel after downsampling, JPEG, and upsampling).

The quantization table that was developed for testing purposes is shown in Figure 14. This table is recommended until optimized quantization tables are generated. The downsampling ratios for this table at several different bit rates are {1.5, 3.0, 6.0, 12.0, 24.0}. The approximate bit rates that correspond to the downsampling ratios are {1/2, 1/4, 1/8, 1/16, 1/32}. These sampling ratios are recommended until the trade-off is further explored. Although five low bit rate quality levels will be defined along with five sets of fixed compression parameters (e.g. downsample ratio, quantization table, and Huffman tables), the downsampling ratio and quantization table should still be left as open parameters. The compression system should still be able to accept arbitrary sampling ratios and different quantization tables. For entropy coding in JPEG, the Quality Level 1 (Q1) Huffman table is recommended. This table provides adequate results for preliminary tests.

36, 36, 37, 39, 42, 45, 50, 54
36, 37, 39, 42, 45, 50, 54, 60
37, 39, 42, 45, 50, 54, 60, 66
39, 42, 45, 50, 54, 60, 66, 74
42, 45, 50, 54, 60, 66, 74, 81
45, 50, 54, 60, 66, 74, 81, 90
50, 54, 60, 66, 74, 81, 90, 99
54, 60, 66, 74, 81, 90, 99, 110

Figure 14: JPEG quantization table for preliminary experiments.